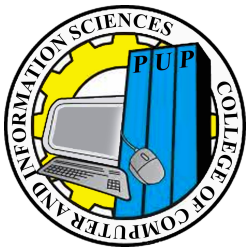
Republic of the Philippines
Polytechnic University of the Philippines
Sta. Mesa, Manila
Department of Computer Science



"VoxTunesAI: Music Player Control Using Speech Recognizer API"

By:

Gallardo, Matthew
Javier, Joyce Marie
Maquiling, Val Jay
Sullos, Kristine Ann

To:

Prof. Ria A. Sagum, MCS, LPT

February 2023

I. Introduction and its Background

In recent years, speech recognition has become increasingly popular in different fields. Speech Recognition converts human sound signals into words or instructions (Liu, 2010). This technology is commonly used to dictate emails, search the internet, and even control specific devices with their voice. However, speech recognition technology also has its limitations. In some situations, it can struggle to understand people with accents or those who speak at a fast pace. In addition, it can be affected by background noise, making it difficult for the computer to transcribe what is being said accurately.

One of the most common uses of speech recognition technology is in voice-activated personal assistants, such as Apple's Siri and Amazon's Alexa. These assistants allow users to interact with their devices simply by speaking commands, allowing for a more natural and intuitive user experience. Speech recognition is also applied in the field of medicine. For example, doctors use speech recognition to transcribe patient notes more quickly and accurately. This can save time and reduce the potential for errors, improving the overall quality of care for patients. Speech recognition technology is also used in the field of education. It helps students with dyslexia and other learning disabilities improve their reading and writing skills. By allowing these students to dictate their thoughts and ideas, speech recognition technology can help them to overcome some of the challenges they face in the classroom. Overall, the use of speech recognition technology is rapidly expanding into a wide range of different fields. It is improving how we interact with technology and making it more accessible to a wider range of people.

As technology continues to evolve, speech recognition is also utilized in music players. Most of the music players that we currently have now require users to navigate menus and select songs using a keyboard or touch screen. It can be difficult for some users, particularly those who struggle with typing or have physical disabilities. In this project, we will develop a music player that utilizes speech recognition technology. We aim to greatly enhance the user experience of music players and help them control their music playback without needing to use physical buttons on the device or a separate device, such as a smartphone.

This paper presents VoxTunes AI, a music player that has speech recognition. The name "Voxtunes AI" is derived from the words "vox," meaning vocal, and "tunes," referring to music. The addition of "AI" signifies that the music player is powered by artificial intelligence. Together, the name represents a music player that uses speech recognition technology. By speaking commands, users can play songs, adjust the volume, and search for a specific song without using a keyboard or touch screen.

II. Related Works

"An Application to Control Media Player with Voice Commands" by Avuçlu, E., Özçifçi, A., & Elen, A. (2020) [1]

The study aims to provide a working computer application that recognizes voice commands to vocally control music player. Their target users are the people that are unable to use music players due to disability. The application used SpeechRecognitionEngine Class in the System.Speech Library in the C# .Net Framework. The application's recognized voice commands are Player, Open, Active, Passive, Play, Pause, Next, Previous and Stop. (Avuçlu, Özçifçi, & Elen, 2020)

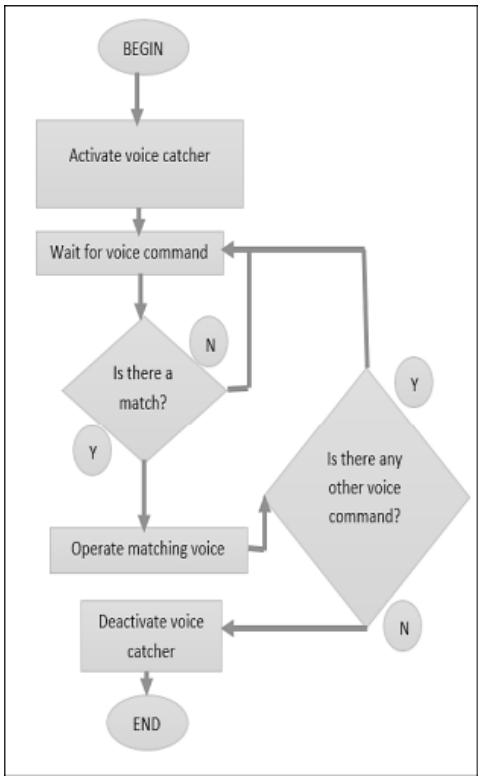


Figure 1: showing the system's flow diagram

III. Objectives of the Project

The research project has the following general and specific objectives.

General objective

The general objective of a music player with speech recognition is to provide a convenient and accessible way for users to control and interact with their music using voice commands. This can improve the user experience and make it easier for people with disabilities to use the music player.

Specific objectives

- Hands-free operation: With speech recognition, users can control their music player without needing to physically touch it, which can be useful while driving or working out.
- Increased accessibility: Speech recognition can make it easier for people with disabilities to use a music player, as they may not be able to use physical controls.
- Convenience: It can be more convenient to use voice commands to control a music player than to navigate menus or use physical buttons
- Improved user experience: A music player with speech recognition can provide a more seamless and enjoyable user experience by allowing users to easily control their music without having to fumble with buttons or menus.

IV. Scope and Limitations of the Project

Researchers aim to work for various commands or functions that a music player will recognize using speech recognition. This voice or speech recognition that is implemented in a music player allows it to understand commands for a song are the following: play, pause, stop, shuffle, control the volume, play next song, play previous song, play random song in the list, and search song in the list.

On the other hand, the music player's voice or speech recognition has its own limitations. The possibility that it is not always supported by any operating system. Additionally, any environment or background noise can affect the result. Also, multiple inputs or speakers including those with an accent or incorrect pronunciation can be impacted, resulting in the system to not transcribe it properly.

V. System Design and Methodology

The application was programmed in Java programming language, specifically Android Studio.

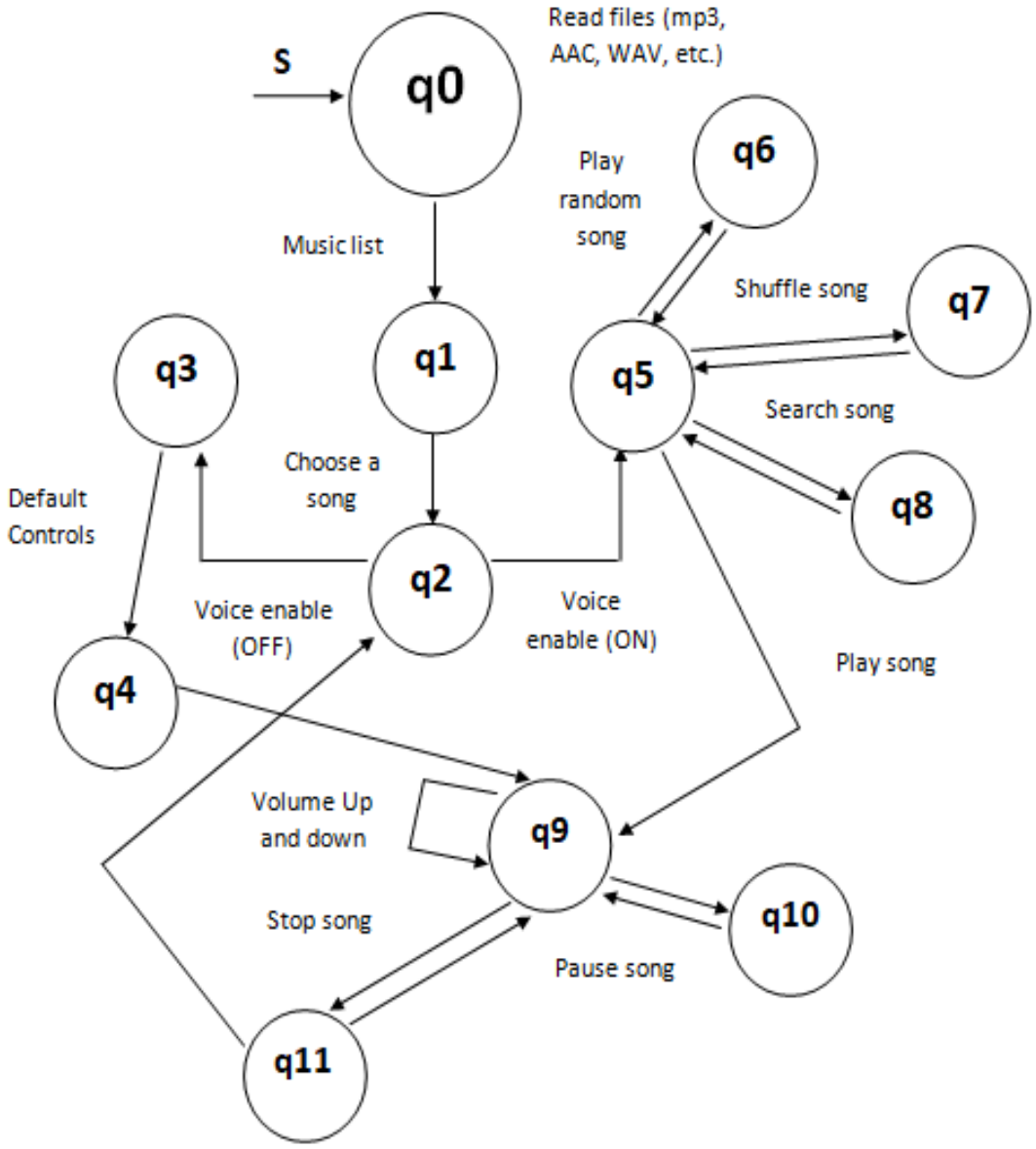


Figure 2: Transition diagram

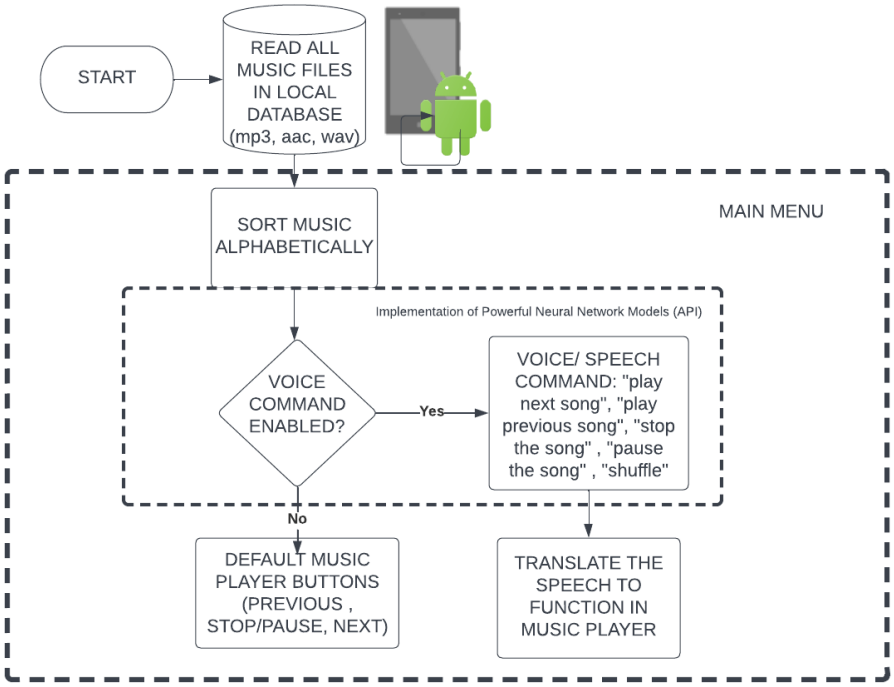


FIGURE 3: SYSTEM DIAGRAM

Speech Recognizer API

The Android Speech Recognizer API is a built-in feature of the Android platform that allows developers to integrate speech recognition functionality into their applications. The API provides a way for developers to capture and transcribe speech input in real-time, and use it for various purposes, such as controlling the application, dictation, or voice search.

The Android Speech Recognizer API uses the Google Cloud Speech-to-Text service to transcribe speech input. It supports multiple languages and can transcribe speech in real-time, allowing users to see the results of their speech input as they speak.

TABLE 1: Command and Functions

Voice Command	**Feature**
Play the song / Play	Mp3 Play
Pause the song/ Pause	Mp3 Pause
Next song/ Next	Mp3 Next song in list
Previous song / Previous	Mp3 Previous song in list
Shuffle song / Shuffle	Mp3 shuffle song in the list
Increase Volume	Mp3 Volume up
Decrease Volume	Mp3 Volume down
Random Song / Random	Play random song in the list
Search for + (Specific Song in Songlist)	Search song on the list

After verification of the required definitions and voice command, the data transmission process is executed with the following code block

```
if (matchesFound != null && !matchesFound.isEmpty()) {
                if (mode.equals("ON")){
                    keeper = matchesFound.get(0);
                    if (keeper.equals("pause the song") || keeper.equals("pause"))
                    {
                        playPauseSong();
                        Toast.makeText(MainActivity.this, "Command" + keeper,
Toast.LENGTH_SHORT).show();

                    }
                    else if (keeper.equals("play the song")|| keeper.equals("play"))
                    {

                        playPauseSong();
                        Toast.makeText(MainActivity.this, "Command recognized" + keeper,
Toast.LENGTH_SHORT).show();
```

```
                }
                else if (keeper.equals("play next song")|| keeper.equals("next"))
                {

                    playNextSong();
                    Toast.makeText(MainActivity.this, "Command recognized" + keeper,
Toast.LENGTH_SHORT).show();

                }
                else if (keeper.equals("play previous song")|| keeper.equals("previous"))
                {

                    playPreviousSong();
                    Toast.makeText(MainActivity.this, "Command recognized" + keeper,
Toast.LENGTH_SHORT).show();

                }
                else if (keeper.equals("increase volume")) {
                    AudioManager audioManager = (AudioManager)
getSystemService(AUDIO_SERVICE);
                    int currentVolume =
audioManager.getStreamVolume(AudioManager.STREAM_MUSIC);
                    audioManager.setStreamVolume(AudioManager.STREAM_MUSIC,
currentVolume + 3, AudioManager.FLAG_SHOW_UI);
                    Toast.makeText(MainActivity.this, "Command recognized: Increase
volume", Toast.LENGTH_SHORT).show();
                }
                else if (keeper.equals("decrease volume")) {
                    AudioManager audioManager = (AudioManager)
getSystemService(AUDIO_SERVICE);
                    int currentVolume =
audioManager.getStreamVolume(AudioManager.STREAM_MUSIC);
                    audioManager.setStreamVolume(AudioManager.STREAM_MUSIC,
currentVolume - 3, AudioManager.FLAG_SHOW_UI);
                    Toast.makeText(MainActivity.this, "Command recognized: Decrease
volume", Toast.LENGTH_SHORT).show();
                }
                else if (keeper.equals("play random song") || keeper.equals("random"))
                {
                    playRandomSong();
                    Toast.makeText(MainActivity.this, "Command recognized: Play random
song", Toast.LENGTH_SHORT).show();
                }
                else if (keeper.contains("search for")) {
                    String keyword = keeper.replace("search for", "");
```

```
                searchAndPlaySong(keyword);
            }

            Toast.makeText(MainActivity.this, "Result " + keeper,
Toast.LENGTH_SHORT).show();
        }
```

VI. Graphical User Interface

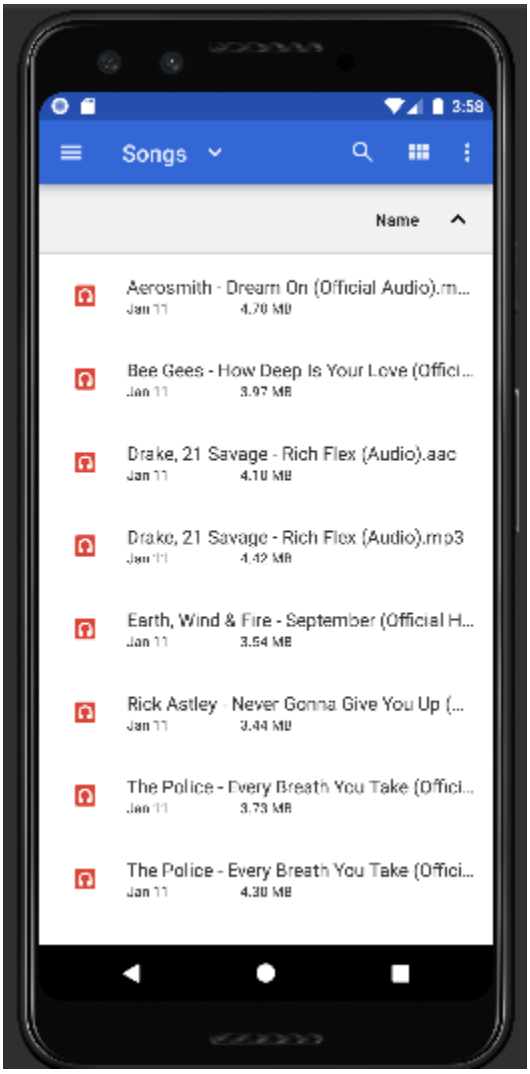


FIGURE 1: LOCAL DATABASE MUSIC FILES

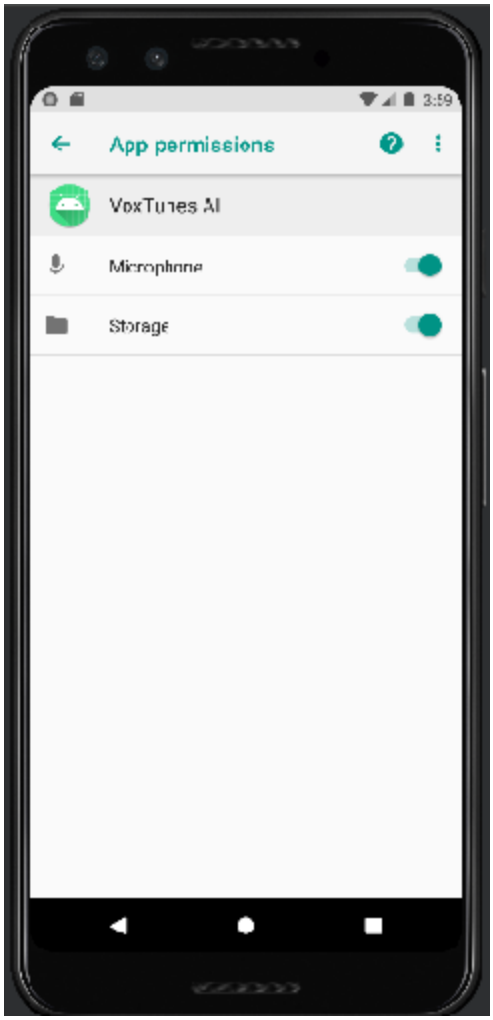


FIGURE 2: APP PERMISSIONS (Microphone and Storage)

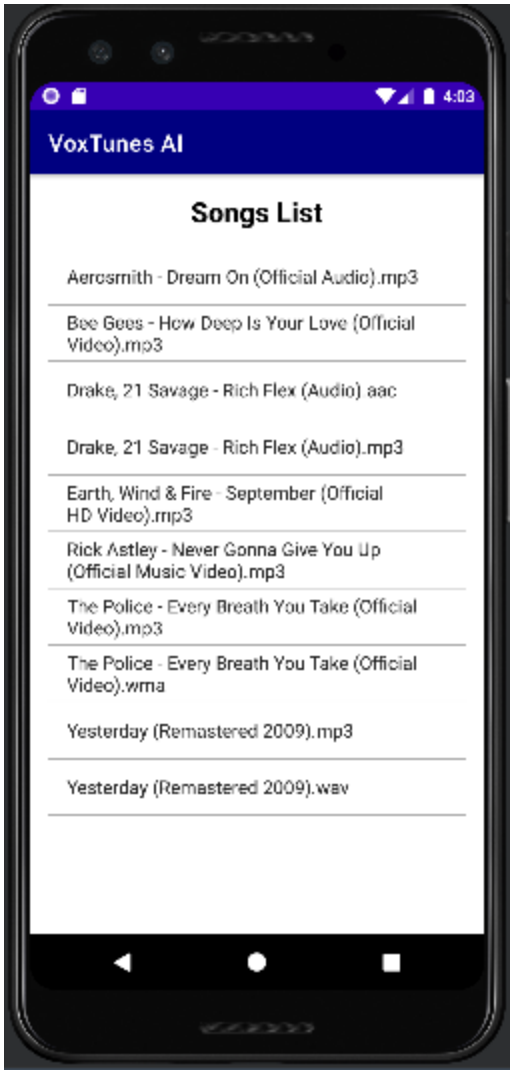


FIGURE 3: SONG LIST

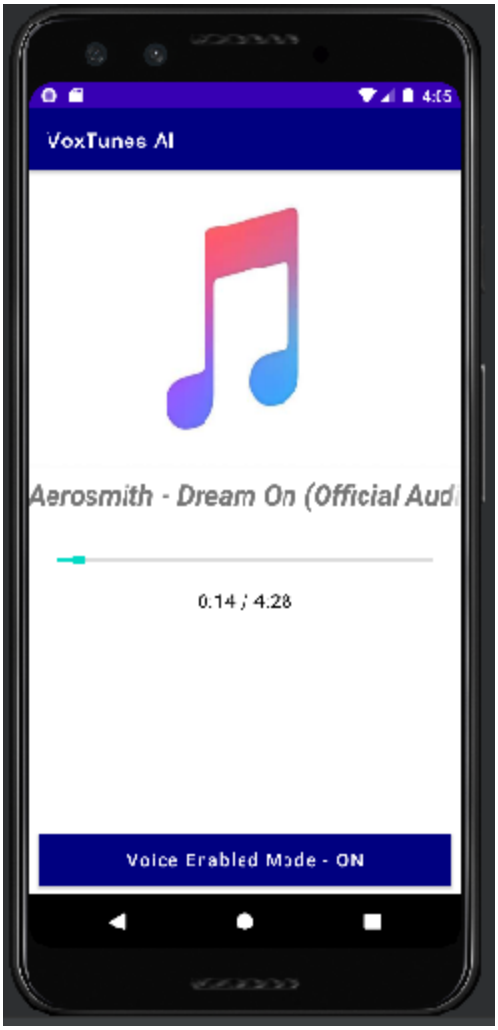


FIGURE 4: VOICE ENABLED MODE-ON INTERFACE

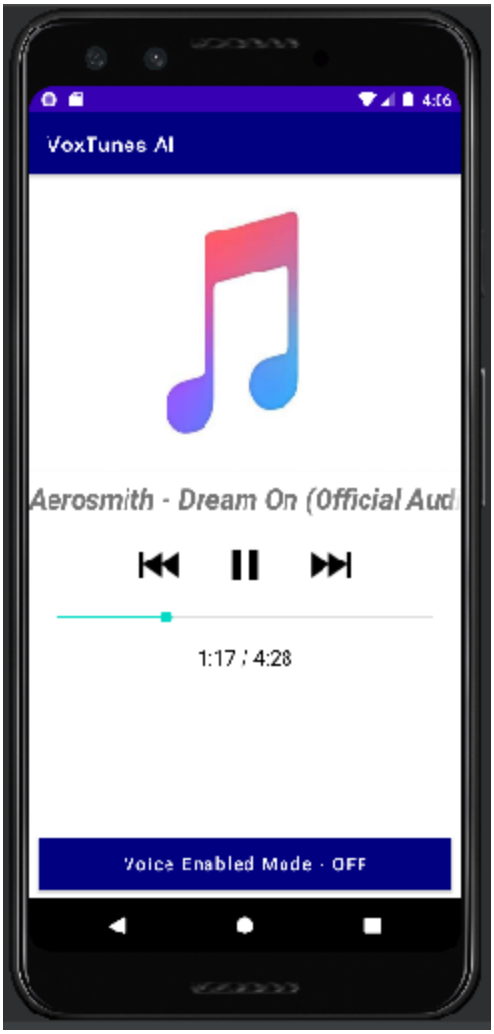
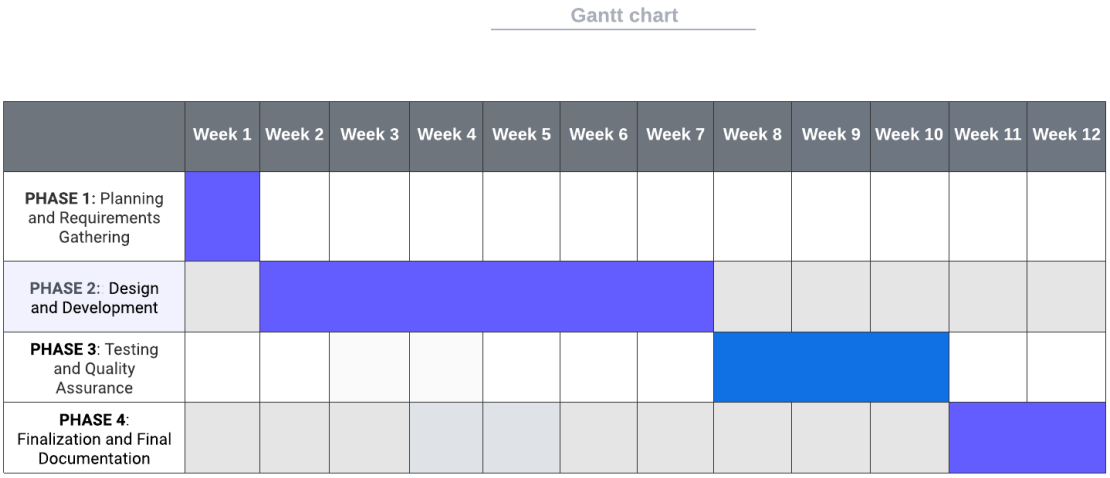


FIGURE 5: VOICE ENABLED MODE-OFF INTERFACE

VII. Schedule of Activities



Phase 1: Planning and Requirements Gathering (1 week)

- Define project goals and objectives
- Identify target audience and user needs
- Conduct market research to understand existing solutions and potential competitors
- Develop project scope and timeline

Phase 2: Design and Development (6 weeks)

- Design user interface and user experience for the music player application
- Develop speech recognition functionality using machine learning algorithms
- Integrate speech recognition functionality into the music player application
- Test and debug the speech recognition system

Phase 3: Testing and Quality Assurance (3 weeks)

- Conduct user testing to gather feedback and identify any issues with the speech recognition system
- Make any necessary changes based on user feedback
- Conduct internal testing to ensure the system meets all functional and performance requirements

Phase 4: Finalization and Final Documentation (2 weeks)

- Finishing touches and preparation to final presentation
- Possible deployment of the mobile app to the public
- Monitor the system for any issues or bugs, and make updates as needed to improve performance and functionality
- Provide ongoing support and maintenance for the system as needed.

VIII. CONCLUSION

Overall, the study found that a music player controlled by a speech recognition API can be a useful tool for individuals who may have difficulty using traditional input devices like a mouse or keyboard. However, the accuracy of the voice recognition is influenced by the difficulty of the pronunciation and length of the voice command.As the pronunciation of the voice command becomes more difficult and the number of letters in it increases, the level of accurate voice recognition decreases. 100% accurate recognition can be achieved by using short words and words with full pronunciation when making voice definitions.

The following in Table 2 shows the results of the experimental studies.

Commands/ Words	Number of Trials	Accurate Recognition	False Recognition	Error Rate
Play	20	19	1	5%
Pause	20	18	2	10%
Previous	20	19	1	5%
Next	20	19	1	5%
Increase Volume	20	17	3	15%
Decrease Volume	20	18	2	10%
Random	20	19	1	10%
Search for Specific Song	20	16	4	20%

As can be seen from the results, it was more difficult to identify words with a high number of words and difficult to pronounce. The VoxTunesAI allows users to perform daily music player functions, such as playing, pausing, stopping, and skipping to the next or previous song, as well as searching for specific songs in the playlist. The application can be controlled remotely by sending voice commands from any location using a wired or wireless headset, without the need for a mouse or keyboard.

This study demonstrates that such a mobile application can be useful for people with disabilities, the elderly, or bedridden patients to meet their listening needs, and can be developed in Java language using the Android Studio IDE. The speech recognition functionality can be implemented using libraries such as Google Speech-to-Text API, providing accurate voice recognition and reducing the error rate. With this mobile application, users can easily manage their music player using voice commands, enabling them to enjoy their music without requiring assistance from others.

IX. References

[1] Avuçlu, E., Özçifçi, A., & Elen, A. (2020). An Application to Control Media Player with Voice Commands. Journal of Polytechnic, Volume 23(Issue 4), page 1311-1315. doi:10.2339/politeknik.646675

[2] M. P. Escalera, J. A. Hernández, F. J. Martínez, and J. Sánchez, "Speech-Based Music Navigation System Using Finite State Automata," in 2006 International Conference on Multimedia and Expo (ICME), 2006

[3] N. R. Iyer, S. S. Patel, and D. D. Patel, "Automata-Based Music Player Control Using Spoken Commands," in 2014 International Conference on Electronics, Communication and Aerospace Technology (ICECA), 2014,

[4] Y. Zhang, X. Li, and Y. Liu, "A Context-Free Grammar-Based Music Player Control System Using Spoken Commands," in 2010 International Conference on Multimedia Technology, 2010,

[5] S. Singh, S. Sahu, G. Kumar and A. Dutta (2018). “API-Based Voice-Assisted Music Player System,” in Extrude Design. Retrieved from https://extrudesign.com/api-based-voice-assisted-music-player-system/

[6] G. Kiruthikamani and E. Esakki Vigneswaran , “Advanced Music Player with Audio Recognition and Touch Interface for Visually Impaired”, (2015).

[7] O. F. Bertol and P. Nohama, "MP3 Player Powered by Voice Commands," 2015 Pan American Health Care Exchanges (PAHCE), 2015, pp. 1-1.

[8] Liu, H. (2020). Introduction. Robot Systems for Rail Transit Applications, 1–36. https://doi.org/10.1016/b978-0-12-822968-2.00001-2